UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 09/621,571 | 07/21/2000 | Rajesh Bordawekar | 13675(YOR9-2000-0365US1 | 4994 |

7590 02/25/2005

Richard L Catania
Scully Scott Murphy & Presser
400 Garden City Plaza
Garden City, NY 11530

| EXAMINER | |
|---|---|
| STEELMAN, MARY J | |

| ART UNIT | PAPER NUMBER |
|---|---|
| 2122 | |

DATE MAILED: 02/25/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

| Office Action Summary | Application No. | Applicant(s) | |
|---|---|---|---|
| | 09/621,571 | BORDAWEKAR ET AL. | |
| | Examiner | Art Unit | |
| | Mary J. Steelman | 2122 | |

*-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --*

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

1)☒ Responsive to communication(s) filed on *1/5/2005, 10/27/2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

4)☒ Claim(s) *31-36* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *31-36* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

**Application Papers**

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

## DETAILED ACTION

1. This Office Action is in response to RCE filed 01/05/2005 and Amendments filed 10/27/2004. Per Applicant's the Specification has been amended. Per Applicant's request, claims 31-34 have been amended. Claims 31-36 are pending.

### *Claim Objections*

2. Claim 31 is unclear to Examiner because of the variations of the terms identifying byte code and code that has been precompiled into an intermediate representation. It is suggested that limitations uniformly reference "byte code representation" (instead of 'the byte code') and "precompiled code" (instead of 'the generated code' or "intermediate...representation') to denote the two versions of program code.

### *Claim Rejections - 35 USC § 101*

3. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

4. Claims 31, and 34-36 are rejected under 35 U.S.C. 101 because the disclosed invention is inoperative and therefore lacks utility. Claim 31 steps f) and g) verify a match, step h) loads and executes, without providing for the possibility of an exception. When verifying the secure hash, there may not be a match of the secure hash of the byte code or generated code with the digitally signed secure hash for the byte code or the generated code, as in steps f and g. Similarly, claim 34, step c) performs a comparison of the secure hash. Step d) interprets byte code without providing for the exception of 'any byte codes associated with any code S relied upon by C (that

may) have changed' as detected by the comparison in step c. In each case, there is no provision for an alternate action / or for restricting the performance of the last step.

### *Claim Rejections - 35 USC § 112*

5. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

6. Claims 31 and 34-36 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. The claims contains subject matter which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention.

Claim 31 steps f) and g) verify a match, step h) loads and executes, without providing for the possibility of an exception. When verifying the secure hash, there may not be a match of the secure hash of the byte code or generated code with the digitally signed secure hash for the byte code or the generated code, as in steps f and g. Similarly, claim 34, step c) performs a comparison of the secure hash. Step d) interprets byte code without providing for the exception of 'any byte codes associated with any code S relied upon by C (that may) have changed' as detected by the comparison in step c. In each case, there is no provision for an alternate action / or for restricting the performance of the last step.

7. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

8. Claims 31 and 32 rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

9. Claim 31 recites "the generated code' in c), d), f), and h). Claim 32 recites "the generated code' in c) and d). Claim 32 recites the limitation "performing the default action" in e). There is insufficient antecedent basis for these limitations in the claims.

### *Claim Rejections - 35 USC § 103*

10. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

11. Claims 31, 33 and 34-36 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent 6,078,744 to Wolczko et al., in view of US Patent 6,367,012 to Atkinson et al., and further in view of US Patent 6,704,927 B1 to Bak et al. (Art made of record.)

Per claim 31:

Wolczko disclosed reuse of saved compilation data (statically precompiled code) that has been journaled. At col. 7, lines 18-22, Wolczko stated, "The record also distinguishes when the compilation unit has changed between the initial and subsequent compilations...many techniques

exist for determining equivalence of the compilation unit." Col. 7, lines 43-53, "The 'compilation unit identification' field contains a value that identifies the compilation unit for the journal record...this value contains verification information used to verify that the compilation unit has not changed between an initial compilation and a subsequent compilation...verification information can be the source code of compilation unit, the checksum of the source code of the compilation unit of similar information." Col. 9, lines 57-60, "...the adaptive compiler is assured that the source program initially compiled is the same as the source program that is optimized." Col. 10, lines 41-66, "As each compilation unit is processed an 'equivalent unit in journal decision procedure determines whether the compilation unit in the source program...is equivalent to the compilation unit processed by the "initial phase compilation' process (by using the verification data stored in the 'information' field)...If the...decision procedure determines that the compilation units are not equivalent or that no information for the compilation unit has been journaled, the 'subsequent phase compilation' (dynamic) (therefore a mixed static and dynamic environment) process continues to 'generate intermediate compilation data' procedure...However, if the...decision procedure determines that the completion units are equivalent...the 'subsequent phase compilation' process continues to 'read ICD data from journal procedure (static) ...instead of computing the ICD." (See Figure 6.) Wolczko disclosed "loading and executing the generated code" at col. 4, lines 41-45, "The JIT compiler compiles a method (a loaded method), a portion of a method, a statement of other source grouping just before the source groupings is first executed (load / execute the generated code). Subsequent calls to the source grouping re-execute the compiled target language for the host computer's ABI."

Wolczko failed to give specifics regarding how the equivalence is determined. However, Atkinson provided more detail on a certification or signature, incorporated in a program, file or code to assure authenticity and integrity, particularly for receiving over a network. (See Figure 3 and col. 6, lines 19-25) Col. 6, lines 30-33, "Code signing method assures the recipient of the identity of the source of file (i.e., its authenticity) (verifying that the intermediate or byte-code representation of the program is safe) and that the file was not modified after it was transmitted by that source (i.e., the integrity of file). Atkinson disclosed hashing (col. 6, lines 40-50), "...a cryptographic digest of "hash" of executable file is obtained (forming a secure hash describing the bye code) or computed (forming a secure hash describing the generated code). Standard hash functions are available...These functions take a...string and convert it to a fixed length output string...(called a cryptographic digest). This... "fingerprints" the file by producing a value that indicates whether a file submitted for download matches the original file (verifying that the secure hash of the byte-code matches the digitally signed secure hash for the byte-code / generated code). Hashing functions and the values they generate are secure..." See Figure 4 and col. 6, line 66 – col. 7, line 8, "...publisher digital certificate (digitally signed secure) and publisher signature are attached or appended to or incorporated to executable file. Publisher signature and publisher digital certificate together form a keyed source confirmation with a secure representation..."

The combination of Wolczko and Atkinson failed to disclose:
"saving pre-compiled programs, including determining where to place said programs, annotating the programs with dependent information, annotating the programs with dependence

information, and processing the programs to product a further annotated executable code with annotations to help adapt the code to a new executable environment;

However, Bak disclosed (col. 2, lines 15-20), "calls may be optimized and dependency information may be added (annotating the programs with dependent information, annotating the programs with dependence information) to the compiled method so that when classes are loaded at runtime execution, it can be determined if the compiled method is still valid, should be interpreted...or recompiled.", (col. 2, lines 27-29) "The dependency information is arranged to indicate a status of the function, and contains information pertaining to the class, the name, and the signature associated with the process.", col. 4, lines 44-48, "A compiled function associated with the system is then inspected. The compiled function includes dependency information that is arranged to indicate a validity status of the compiled function as well as the optimization status of the compiled function (processing the programs to product a further annotated executable code with annotations to help adapt the code to a new executable environment)", (col. 4, lines 53-54), "a determination is made regarding whether the compiled function is invalid", (col. 6, line 61- col. 7, line 12), "the system adds dependency information the compiled method. The dependency information may include the class, function name, and signature...dependency information may be checked for a compiled method to determine if the compiled method is still valid...A compiled method includes a header and compiled code. Header includes among other things...dependency information...Although this information may be stored as a simple list (saving pre-compiled programs), a variety of other storage techniques may be utilized."

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Wolczko's invention that verifies that subsequent compilations

have not changed, by specifying techniques such as those disclosed by Atkinson using secure hashing, digitally signing, and attaching a publisher signature and publisher digital certificate as methods to assure authenticity and integrity, because these are well known, secure ways to determine equivalency of files, an important feature for allowing code generated by the processes to be exchanged and trusted between machines. And to further modify the Wolczko / Atkinson combination, by using Bak's invention that annotates the pre-compiled code with dependent and dependence information, processing the code to produce executable code with annotations to help adapt the code to a new executable environment because all inventions refer to pre-compiling code with consideration of execution performance, while ensuring validity of the pre-compiled code, thereby producing a more efficient and correct execution.

Per claim 33:

Wolczko disclosed a mixed static and dynamic environment, col. 10, lines 41-66, "As each compilation unit is processed an 'equivalent unit in journal decision procedure determines whether the compilation unit in the source program...is equivalent to the compilation unit processed by the "initial phase compilation' (statically precompiled code) process (by using the verification data stored in the 'information' field)...If the...decision procedure determines that the compilation units are not equivalent or that no information for the compilation unit has been journaled, the 'subsequent phase compilation' (dynamic / updating statically generated code) (therefore a mixed static and dynamic environment) process continues to 'generate intermediate compilation data' (compiler generating the code for S (separately compiled code) to a) associate with method and data names, or signatures)" At col. 7, lines 18-22, Wolczko stated, "The record

also distinguishes when the compilation unit has changed between the initial and subsequent compilations (- c) check if byte codes associated with any names in the S (separately compiled code) relied upon by C (statically generated code) have changed by comparing)...many techniques exist for determining equivalence of the compilation unit." Col. 7, lines 43-53, "The 'compilation unit identification' field contains a value that identifies the compilation unit for the journal record...this value contains verification information used to verify that the compilation unit has not changed between an initial compilation and a subsequent compilation...verification information can be the source code of compilation unit, the checksum of the source code of the compilation unit of similar information." Col. 9, lines 57-60, "...the adaptive compiler is assured that the source program initially compiled is the same as the source program that is optimized." Col. 10, lines 41-66, "As each compilation unit is processed an 'equivalent unit in journal decision procedure determines whether the compilation unit in the source program...is equivalent to the compilation unit processed by the "initial phase compilation' process (by using the verification data stored in the 'information' field)...If the...decision procedure determines that the compilation units are not equivalent or that no information for the compilation unit has been journaled, the 'subsequent phase compilation' (- d)dynamically recompile the byte codes associated with C if any byte codes associated with S have changed) process continues to 'generate intermediate compilation data' procedure...However, if the...decision procedure determines that the completion units are equivalent...the 'subsequent phase compilation' process continues to 'read ICD data from journal procedure (static) ...instead of computing the ICD." (See Figure 6.) Wolczko disclosed "loading and executing the generated code" at col. 4, lines 41-45, "The JIT compiler compiles a method (a loaded method), a portion of a method, a

statement of other source grouping just before the source groupings is first executed (executing the generated code). Subsequent calls to the source grouping re-execute the compiled target language for the host computer's ABI."

Wolczko failed to give specifics regarding how the equivalence is determined. However, Atkinson provided more detail on a "secure hash of the name of the method and data names or signatures in S with the compiler for C recording the secure hash for the hash code". Atkinson disclosed hashing (col. 6, lines 40-50), "...a cryptographic digest of "hash" of executable file is obtained or computed (compiler associates with method and data names, or signatures, in S a secure hash, recording the secure hash). Standard hash functions are available...These functions take a...string and convert it to a fixed length output string...This... "fingerprints" the file by producing a value that indicates whether a file submitted for download matches the original file (comparing the secure hash of the names associated with S with the secure hash stored for this byte code in C). Hashing functions and the values they generate are secure..."

The combination of Wolczko and Atkinson failed to disclose:
"saving pre-compiled programs, including determining where to place said programs, annotating the programs with dependent information, annotating the programs with dependence information, and processing the programs to product a further annotated executable code with annotations to help adapt the code to a new executable environment."

However, Bak disclosed (col. 2, lines 15-20), "calls may be optimized and dependency information may be added (annotating the programs with dependent information, annotating the programs with dependence information) to the compiled method so that when classes are loaded at runtime execution, it can be determined if the compiled method is still valid, should be

interpreted...or recompiled.", (col. 2, lines 27-29) "The dependency information is arranged to indicate a status of the function, and contains information pertaining to the class, the name, and the signature associated with the process.", col. 4, lines 44-48, "A compiled function associated with the system is then inspected. The compiled function includes dependency information that is arranged to indicate a validity status of the compiled function as well as the optimization status of the compiled function (processing the programs to product a further annotated executable code with annotations to help adapt the code to a new executable environment)", (col. 4, lines 53-54), "a determination is made regarding whether the compiled function is invalid", (col. 6, line 61- col. 7, line 12), "the system adds dependency information the compiled method. The dependency information may include the class, function name, and signature...dependency information may be checked for a compiled method to determine if the compiled method is still valid...A compiled method includes a header and compiled code. Header includes among other things...dependency information...Although this information may be stored as a simple list (saving pre-compiled programs), a variety of other storage techniques may be utilized."

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Wolczko's invention that verifies that subsequent compilations have not changed, by specifying techniques such as those disclosed by Atkinson using secure hashing, digitally signing, and attaching a publisher signature and publisher digital certificate as methods to assure authenticity and integrity, because these are well known, secure ways to determine equivalency of files, an important feature for allowing code generated by the processes to be exchanged and trusted between machines. And to further modify the Wolczko / Atkinson combination, by using Bak's invention that annotates the pre-compiled code with dependent and

dependence information, processing the code to produce executable code with annotations to help adapt the code to a new executable environment because all inventions refer to pre-compiling code with consideration of execution performance, while ensuring validity of the pre-compiled code, thereby producing a more efficient and correct execution.

Per claim 34:

Wolczko disclosed a mixed static and dynamic environment, col. 10, lines 41-66, "As each compilation unit is processed an 'equivalent unit in journal decision procedure determines whether the compilation unit in the source program...is equivalent to the compilation unit processed by the "initial phase compilation' (statically precompiled code) process (by using the verification data stored in the 'information' field)...If the...decision procedure determines that the compilation units are not equivalent or that no information for the compilation unit has been journaled, the 'subsequent phase compilation' (separately compiled ) (therefore a mixed static and dynamic environment) process continues to 'generate intermediate compilation data'. (use of statically generated code for some byte code that depends on some byte code that may be separately compiled)

Wolczko disclosed reuse of saved compilation data (statically generated code) that has been journaled. At col. 7, lines 18-22, Wolczko stated, "The record also distinguishes when the compilation unit has changed between the initial and subsequent compilations...many techniques exist for determining equivalence of the compilation unit." (check if any byte codes associated with any code S relied upon by C have changed) Col. 7, lines 43-53, "The 'compilation unit identification' field contains a value that identifies the compilation unit for the journal

record...this value contains verification information used to verify that the compilation unit has not changed between an initial compilation and a subsequent compilation...verification information can be the source code of compilation unit, the checksum of the source code of the compilation unit of similar information." Col. 10, lines 41-66, "As each compilation unit is processed an 'equivalent unit in journal decision procedure determines whether the compilation unit in the source program...is equivalent to the compilation unit processed by the "initial phase compilation' process. Wolczko disclosed "interpreting the byte codes corresponding to C (statically generated code)" at col. 4, lines 41-45, "Subsequent calls to the source grouping re-execute (interpret the byte codes corresponding to C) the compiled target language for the host computer's ABI."

Wolczko failed to give specifics regarding the use of a secure hash. However, Atkinson provided more detail on "associating with S a secure hash of the byte code associated with S" and "secure hash for the byte code corresponding to any S that affects the code generated for C". Atkinson disclosed hashing (col. 6, lines 40-50), "...a cryptographic digest of "hash" of executable file is obtained or computed (associating with S a secure hash of the byte code). Standard hash functions are available...These functions take a...string and convert it to a fixed length output string.... This... "fingerprints" the file by producing a value that indicates whether a file submitted for download matches the original file (secure hash for byte code corresponding to any S that affects the code generated for C). Hashing functions and the values they generate are secure..."

The combination of Wolczko and Atkinson failed to disclose:

"saving pre-compiled programs, including determining where to place said programs, annotating the programs with dependent information, annotating the programs with dependence information, and processing the programs to product a further annotated executable code with annotations to help adapt the code to a new executable environment."

However, Bak disclosed (col. 2, lines 15-20), "calls may be optimized and dependency information may be added (annotating the programs with dependent information, annotating the programs with dependence information) to the compiled method so that when classes are loaded at runtime execution, it can be determined if the compiled method is still valid, should be interpreted...or recompiled.", (col. 2, lines 27-29) "The dependency information is arranged to indicate a status of the function, and contains information pertaining to the class, the name, and the signature associated with the process.", col. 4, lines 44-48, "A compiled function associated with the system is then inspected. The compiled function includes dependency information that is arranged to indicate a validity status of the compiled function as well as the optimization status of the compiled function (processing the programs to product a further annotated executable code with annotations to help adapt the code to a new executable environment)", (col. 4, lines 53-54), "a determination is made regarding whether the compiled function is invalid", (col. 6, line 61- col. 7, line 12), "the system adds dependency information the compiled method. The dependency information may include the class, function name, and signature...dependency information may be checked for a compiled method to determine if the compiled method is still valid...A compiled method includes a header and compiled code. Header includes among other things...dependency information...Although this information may be stored as a simple list (saving pre-compiled programs), a variety of other storage techniques may be utilized."

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Wolczko's invention that verifies that subsequent compilations have not changed, by specifying techniques such as those disclosed by Atkinson using secure hashing, digitally signing, and attaching a publisher signature and publisher digital certificate as methods to assure authenticity and integrity, because these are well known, secure ways to determine equivalency of files, an important feature for allowing code generated by the processes to be exchanged and trusted between machines. And to further modify the Wolczko / Atkinson combination, by using Bak's invention that annotates the pre-compiled code with dependent and dependence information, processing the code to produce executable code with annotations to help adapt the code to a new executable environment because all inventions refer to pre-compiling code with consideration of execution performance, while ensuring validity of the pre-compiled code, thereby producing a more efficient and correct execution.

Per claim 35:

-the compiler performing dependence checks during program execution to avoid using a stale code for a procedure in the event of changes to other codes.

Wolczko disclosed 'dependence checks' at col. 10, lines 41-66. "As each compilation unit is processed an 'equivalent unit in journal' decision procedure (dependence check) determines whether the compilation unit in the source program, currently being compiled, is equivalent to the compilation unit processed by the 'initial phase compilation' (precompiled code) process..."

Per claim 36:

-the step of performing dependence checks includes using time stamps to determine if any of the classes on which the code is dependent have changed.

Wolczko disclosed (col. 11, lines 8-15), "...determination made by the 'journal ICD' decision procedure and the 'equivalent unit in journal' decision procedure are dependent on the processing speed...and other aspects of the computer executing the compiler. Those skilled in the art will also understand that these decisions can be heuristically programmed based on instrumenting..."

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to include 'time stamps' in determining whether classes on which the code id dependent has changed. This is well known in the art. Updated time stamps can easily be used for the basis of heuristic programming.

12. Claim 32 is rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent 6,078,744 to Wolczko et al., in view of US Patent 6,704,927 B1 to Bak.

Wolczko disclosed a mixed static and dynamic environment (col. 10, lines 41-66) that provides statically precompiled code to a virtual machine (col. 9, lines 22-33), whereas the virtual machine would JIT compile code dynamically if any change was detected. Wolczko did not disclose using the compiler for -maintaining symbolic entries for externally referenced

symbols; -maintaining a mapping from locations in the generated code that reference external symbols to the symbolic entry for that symbol and the virtual machine, before the code is executed, performs the steps of -using the mapping and symbolic entries created by the compiler to generate direct references in the generated code to the externally referenced symbols that have been resolved by the virtual machine; -performing the default action on those external symbols that have not been resolved.

However, Bak disclosed (col. 6, lines 8-25), "Each Java class has a constant pool associated with it. The constant pool is stored...and serves a function similar to symbol tables...each entry...is indexed. In addition to the constant pool storing literal constants, the constant pool stores classes, methods, fields, and interfaces symbolically (maintaining symbolic entries for externally referenced symbols)...By storing names and not address, the Java runtime system resolves (maintaining mapping) the symbolic reference into a physical address dynamically at runtime (generate direct references).

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Wolczko's invention used in a mixed static and dynamic environment to include features as disclosed by Bak, by using mapping and symbolic entries to generate direct references when resolving at runtime, because generating direct references result in "load immediate" instructions are which superior, quicker processes.

### *Response to Arguments*

11. Applicant's arguments with respect to claims 31-34 have been considered but are moot in view of the new grounds of rejection, US Patent 6,704,927 B1 to Bak et a..

***Conclusion***

12. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Mary Steelman, whose telephone number is (571) 272-3704. The examiner can normally be reached Monday through Thursday, from 7:00 AM to 5:30 PM If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached at (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Mary Steelman

02/17/2005

TUAN DAM
SUPERVISORY PATENT EXAMINER